

Лабораторна робота № 2

Створення власних компонентів у Platform Designer Tool

При розробці власної системи у Platform Designer Tool може статися так, що серед стандартних компонентів буде відсутнім потрібний вам. Тому, варто опанувати можливість створювати власні компоненти, яку надає Platform Designer Tool.

У даному описі розглянуто роботу у Quartus Prime 18.1 Lite із платою Terasic DE1-SoC.



Корисні посилання:

Making Platform Designer Components

ftp://ftp.intel.com/Pub/fpgaup/pub/Teaching_Materials/current/Tutorials/Making_Qsys_Components.pdf

Avalon Interconnect Specifications

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/manual/mnl_avalon_spec.pdf

Intel FPGA Academic Program Tutorials

<https://software.intel.com/en-us/fpga-academic/learn/tutorials>

1. Вступ.

Platform Designer component — це електронний пристрій, який доступний як бібліотечний компонент для використання у Platform Designer Tool. Вони, зазвичай, складаються з двох частин: внутрішньої - яка містить опис принципу дії, та зовнішньої — яка містить опис інтерфейсу до шини Avalon.

Основні типи інтерфейсів шини Avalon:

- 1) **Avalon Clock Interface** – розповсюдження тактових сигналів;
- 2) **Avalon Reset Interface** – розповсюдження сигналу скидання;
- 3) **Avalon Memory-Mapped Interface (Avalon MM)** – інтерфейс, що забезпечує виконання операцій зчитування-запису за адресами пристроїв. Такий інтерфейс є типовим для архітектури **master - slave**.
- 4) **Avalon Streaming Interface (Avalon-ST)** – інтерфейс, що використовує односпрямований потік даних (передавач – приймач);
- 5) **Avalon Conduit Interface** – об'єднує усі користувацькі інтерфейси, які не належать до інших типів. Можуть використовуватися для сполучень за межі системи, що розробляється у Platform Designer (з зовнішніми компонентами).

Будь-який компонент може підтримувати одразу кілька інтерфейсів різних типів, але обов'язково він повинен мати **Clock** і **Reset** інтерфейси.

У якості прикладу в даній роботі розглянемо створення дешифратора для семисегментного індикатора. Окрім обов'язкових сигналів (пов'язаних з формуванням символів на індикаторі), дешифратор буде мати **Memory-Mapped** та **Conduit** інтерфейси.

Нижче наведено код мовою Verilog, що описує 32-бітний регістр (**reg32**) для запису значення, яке має відображатися на індикаторі та модуль **Memory-Mapped** інтерфейсу (**reg32_avalon_interface**)

для сполучення регістру з шиною Avalon. Тобто, наведений код описує 2 модуля: власне регістр та блок інтерфейсу регістра до шини Avalon. Особливістю шини Avalon є можливість вести запис до регістру кожного байту окремо, для цього використовується 4-бітний сигнал *byteenable*.

```
module reg32 (clock, resetn, D, byteenable, Q);  
  input clock, resetn;  
  input [3:0] byteenable;  
  input [31:0] D;  
  output reg [31:0] Q;  
  always @(posedge clock)  
    if (!resetn)  
      Q <= 32'b0;  
    else  
  begin  
    // Enable writing to each byte separately  
    if (byteenable[0]) Q[7:0] <= D[7:0];  
    if (byteenable[1]) Q[15:8] <= D[15:8];  
    if (byteenable[2]) Q[23:16] <= D[23:16];  
    if (byteenable[3]) Q[31:24] <= D[31:24];  
  end  
endmodule
```

```
module reg32_avalon_interface (clock, resetn, writedata,  
  readdata, write, read,  
  byteenable, chipselect, Q_export);  
  // signals for connecting to the Avalon fabric  
  input clock, resetn, read, write, chipselect;  
  input [3:0] byteenable;  
  input [31:0] writedata;  
  output [31:0] readdata;  
  // signal for exporting register contents  
  // outside of the embedded system  
  output [31:0] Q_export;
```

```

wire [3:0] local_byteenable;
wire [31:0] to_reg, from_reg;
assign to_reg = writedata;
assign local_byteenable = (chipselct & write) ?
    byteenable : 4'd0;
reg32 U1 ( .clock(clock), .resetn(resetn), .D(to_reg),
    .byteenable(local_byteenable), .Q(from_reg) );
assign readdata = from_reg;
assign Q_export = from_reg;
endmodule

```

Для детального розуміння процедури передачі даних рекомендуємо познайомитись з описом *Avalon Interconnect Specifications*.

2. Порядок виконання роботи.

1. Створіть новий проект у Quartus. Назвіть його *component_tutorial*. Виконайте всі потрібні налаштування.
2. Додайте до проекту файли *reg32.v* та *reg32_avalon_interface.v*, з кодом, наведеним вище.
3. Відкрийте *Platform Designer Tool*.
4. Додайте компоненти *Nios II (Classic) Processor* та *On-Chip Memory (RAM or ROM) Intel FPGA IP*. Сполучіть їх як зображено на рис. 1 та налаштуйте як у попередній лабораторній роботі. Виконайте автоматичне призначення адреси.

Use	Connections	Name	Description
<input checked="" type="checkbox"/>		clk_0	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II (Classic) Processor
		clk	Clock Input
		reset_n	Reset Input
		data_master	Avalon Memory Mapped Master
		instruction_master	Avalon Memory Mapped Master
		d_irq	Interrupt Receiver
		jtag_debug_module...	Reset Output
		jtag_debug_module	Avalon Memory Mapped Slave
		custom_instructio...	Custom Instruction Master
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)
		clk1	Clock Input
		s1	Avalon Memory Mapped Slave
		reset1	Reset Input

Рисунок 1. Сполучення компонентів.

5. Наразі необхідно створити власний компонент, який буде додано до процесорного ядра Nios II. Натисніть **New Component** на вкладці *IP Catalog* (рис. 2). Відкриється вікно **Component Editor**, заповніть його у відповідності до рис. 3.

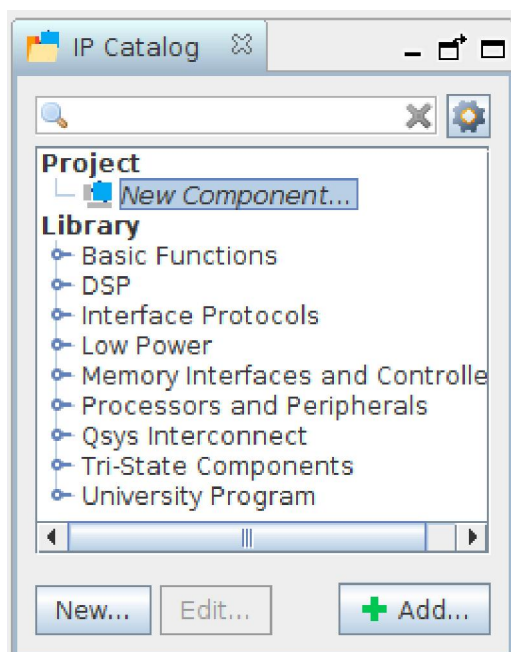


Рисунок 2. Вкладка *IP Catalog*.

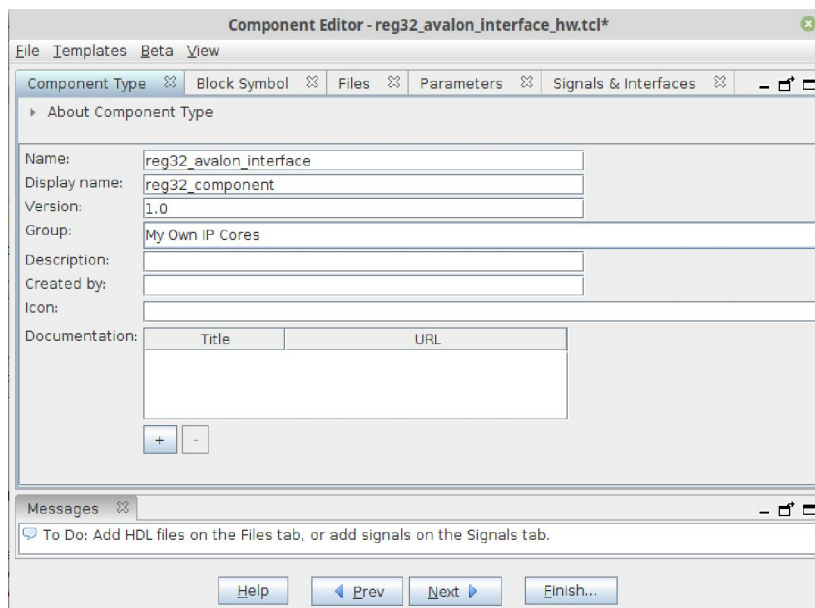


Рисунок 3. Вікно *Component Editor*.

6. Наступний крок – додати файли з описом компонента. Перейдіть до вкладки **Files**. Додайте **reg32_avalon_interface.v**. Натисніть кнопку **Analyze Synthesis Files**. Цей аналіз потрібен для визначення відсутності помилок у обраному файлі.

7. Також, додайте файл **reg32.v** та проаналізуйте його. Якщо аналіз виявить помилки, виправте їх і проведіть аналіз повторно.

8. Натисніть кнопку **Copy from Synthesis Files** в розділі **Verilog Simulation Files**. Це вказівка компілятору обирати дані для моделювання з синтезованого файлу.

9. Далі необхідно узгодити інтерфейс власного компонента з інтерфейсом шини Avalon. Перейдіть на вкладку **Signals & Interfaces**. Зліва, натисніть на полі **<<add interface>>** і оберіть з контекстного меню **Clock Input**. Перетягніть сигнал **clock[1]** до новоствореного інтерфейсу та змініть тип сигналу (у правому вікні) на **clk**. Результат зображено на рис. 4.

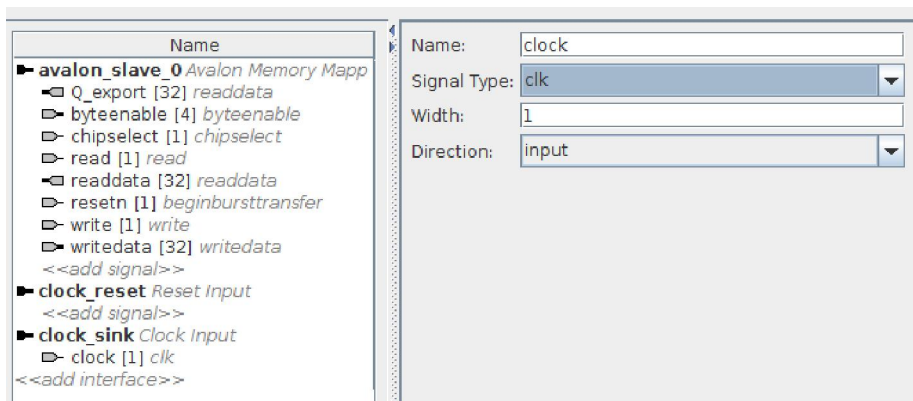


Рисунок 4. Налаштування сигналу **clock[1]**.

10. Виконайте відповідні налаштування для інших сигналів. Результат повинен відповідати рис. 5.

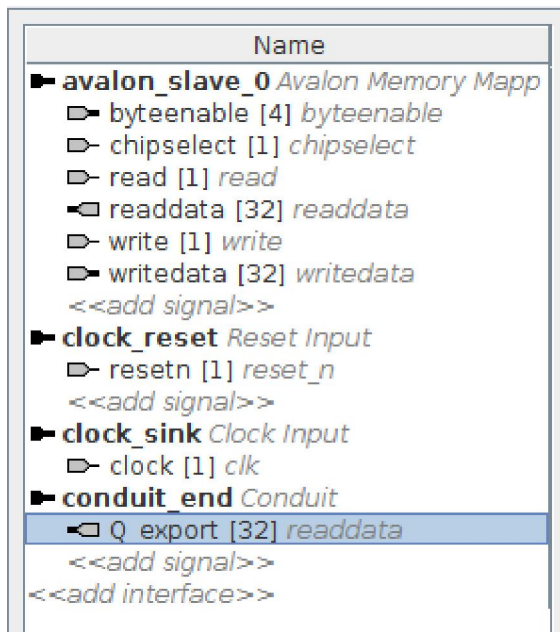


Рисунок 5. Налаштування сигналів.

11. Тепер перейдіть до *avalon_slave_0* (виділіть шину) та у правому вікні зробіть наступні налаштування:

Associated Clock = *clock_sink*

Associated Reset = *clock_reset*

Read Wait = 0

Інші налаштування залиште за замовчанням.

12. Виконайте наступні налаштування для шини *clock_reset*:

Associated Clock = *clock_sink*

Натисніть ***Finish***.

13. Збережіть компонент, коли виникне відповідний запит.

14. Ви повернулися до головного вікна ***Platform Designer***. Додайте щойно створений компонент. При додаванні відкриється вікно, зображене на рис. 6. Натисніть ***Finish***.

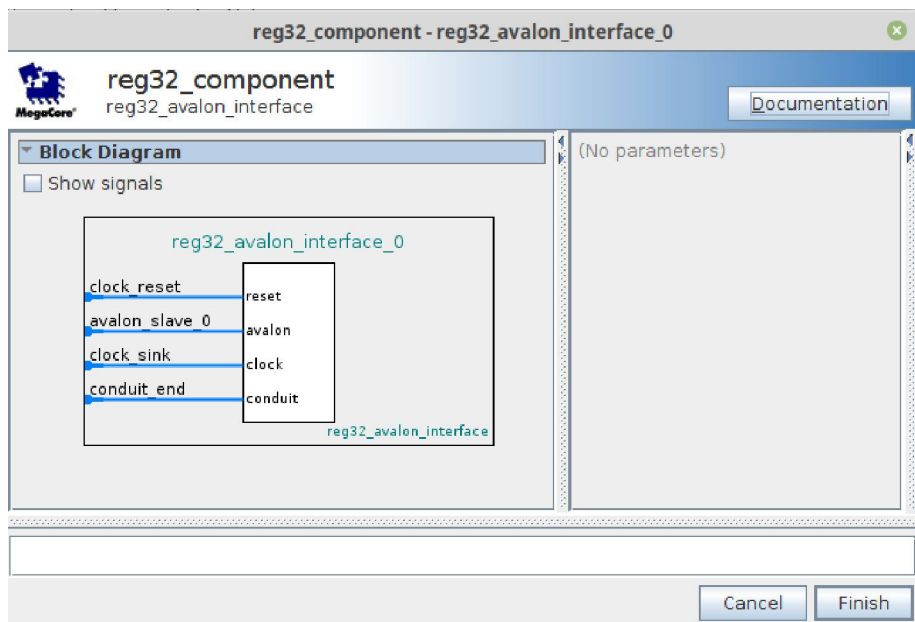


Рисунок 6. Додання до проекту *reg32_component*.

17. Підключіть власний компонент як зображено на рис. 7. Встановіть базову адресу *reg32_avalon_interface_0* рівною 0, іншим компонентам призначте її автоматично. Для *reg32_avalon_interface_0* → *Conduit_end* у графу *export* впишіть *to_hex*.

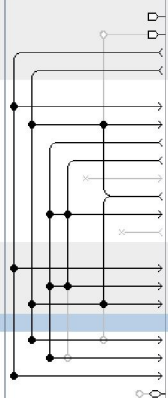
Use	Connections	Name	Description	Export	Clock	Base
<input checked="" type="checkbox"/>		<div>clk_0</div> <div>clk_in</div> <div>clk_in_reset</div> <div>clk</div> <div>clk_reset</div>	<div>Clock Source</div> <div>Clock Input</div> <div>Reset Input</div> <div>Clock Output</div> <div>Reset Output</div>	<div>clk</div> <div>reset</div> <div>Double-click</div> <div>Double-click</div>	<div>exported</div> <div>clk_0</div>	
<input checked="" type="checkbox"/>		<div>nios2_qsys_0</div> <div>clk</div> <div>reset_n</div> <div>data_master</div> <div>instruction_master</div> <div>d_irq</div> <div>jtag_debug_module_reset</div> <div>jtag_debug_module</div> <div>custom_instruction_ma...</div>	<div>Nios II (Classic) Processor</div> <div>Clock Input</div> <div>Reset Input</div> <div>Avalon Memory Mapped Master</div> <div>Avalon Memory Mapped Master</div> <div>Interrupt Receiver</div> <div>Reset Output</div> <div>Avalon Memory Mapped Slave</div> <div>Custom Instruction Master</div>	<div>Double-click</div> <div>Double-click</div> <div>Double-click</div> <div>Double-click</div> <div>Double-click</div> <div>Double-click</div> <div>Double-click</div>	<div>clk_0</div> <div>[clk]</div> <div>[clk]</div> <div>[clk]</div> <div>[clk]</div> <div>[clk]</div>	<div>IRQ 0</div> <div>0x3000</div>
<input checked="" type="checkbox"/>		<div>onchip_memory2_0</div> <div>clk1</div> <div>s1</div> <div>reset1</div>	<div>On-Chip Memory (RAM or ROM...)</div> <div>Clock Input</div> <div>Avalon Memory Mapped Slave</div> <div>Reset Input</div>	<div>Double-click</div> <div>Double-click</div> <div>Double-click</div>	<div>clk_0</div> <div>[clk1]</div> <div>[clk1]</div>	<div>0x2000</div>
<input checked="" type="checkbox"/>		<div>reg32_avalon_interfac...</div> <div>reg32_component</div> <div>clock_reset</div> <div>avalon_slave_0</div> <div>clock_sink</div> <div>conduit_end</div>	<div>reg32_component</div> <div>Reset Input</div> <div>Avalon Memory Mapped Slave</div> <div>Clock Input</div> <div>Conduit</div>	<div>Double-click</div> <div>Double-click</div> <div>Double-click</div> <div>to_hex</div>	<div>[clock_s...</div> <div>[clock_s...</div> <div>clk_0</div> <div>[clock_s...</div>	<div>0x0000</div>

Рисунок 7. Сполучення компонентів.

18. Збережіть створену систему під назвою *embedded_system*.
Оберіть меню **Generate** → **Show Instantiation Template**. Ви побачите вікно, показане на рис. 8.

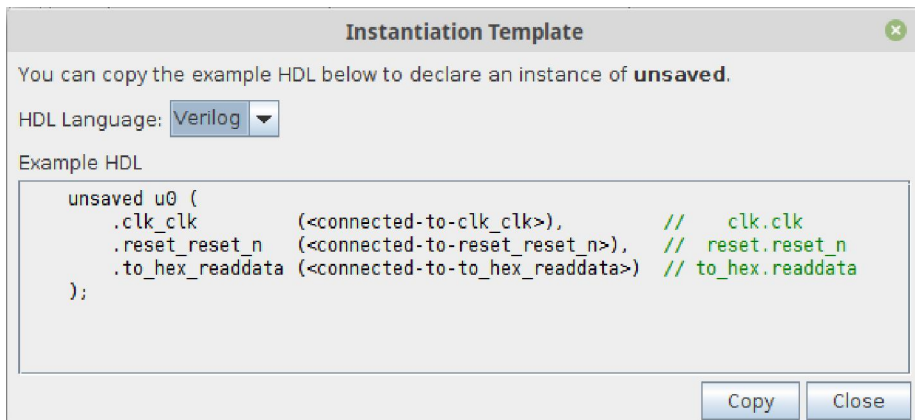


Рисунок 8. Шаблон для використання системи.

19. Згенеруйте систему і закрийте *Platform Designer*.

20. У Quartus Prime додайте до проекту файл *embedded_system/synthesis/embedded_system.qip* (згенерована система).

21. Далі створіть 2 файли: **component_tutorial.v** та **hex7seg.v**. Помістіть до них наступний код та додайте їх до проекту:

```
module component_tutorial (CLOCK_50, KEY,
                          HEX0, HEX1, HEX2, HEX3);
    input CLOCK_50;
    input [0:0] KEY;
    output [0:6] HEX0, HEX1, HEX2, HEX3;
    wire [15:0] to_HEX;

    embedded_system U0 (
        .clk_clk(CLOCK_50),
        .reset_reset_n(KEY[0]),
        .to_hex_readdata(to_HEX)
    );

    hex7seg h0(to_HEX[3:0], HEX0);
    hex7seg h1(to_HEX[7:4], HEX1);
    hex7seg h2(to_HEX[11:8], HEX2);
endmodule
```

```
module hex7seg (hex, display);
    input [3:0] hex;
    output [0:6] display;
    reg [0:6] display;

    /*
     *  - 0 -
     * 5 |   | 1
     *  - 6 -
     * 4 |   | 2
     *  - 3 -
     */

    always @ (hex)
```

case (hex)

```
4'h0: display = 7'b0000001;  
4'h1: display = 7'b1001111;  
4'h2: display = 7'b0010010;  
4'h3: display = 7'b0000110;  
4'h4: display = 7'b1001100;  
4'h5: display = 7'b0100100;  
4'h6: display = 7'b0100000;  
4'h7: display = 7'b0001111;  
4'h8: display = 7'b0000000;  
4'h9: display = 7'b0001100;  
4'hA: display = 7'b0001000;  
4'hb: display = 7'b1100000;  
4'hC: display = 7'b0110001;  
4'hd: display = 7'b1000010;  
4'hE: display = 7'b0110000;  
4'hF: display = 7'b0111000;
```

endcase

endmodule

22. Виконайте призначення контактів (з файлу DE1-SoC.qsf).

Запустіть компіляцію проекту.

23. Відкрийте **Intel FPGA Monitor Program**. Створіть новий проект. Для 2-х перших вкладок вкажіть дані як у попередній лабораторній роботі, а на вкладці **Specify a program type** вкажіть **No Program**.

24. Після створення проекту, оберіть меню **Actions** → **Connect to System**.

25. Перейдіть до вкладки **Memory Tab** та натисніть **Refresh**.

Оберіть адресу **0x0000**. Впишіть у цю комірку значення **0x1234579d**. Спостерігайте результат на платі.

3. Самостійна робота

Завдання 1. Доопрацюйте проект таким чином, щоб були задіяні всі шість семи сегментних індикатори, які розташовані на платі DE1-SoC. Продемонструйте роботу пристрою.

Завдання 2. Доопрацюйте проект таким чином, щоби була можливість програмного керування відображенням значення на семи сегментних індикаторах. Значення повинно задавати за допомогою перемикачів на платі DE1-SoC. Продемонструйте роботу пристрою.